
Lernpaket
Experimente
mit USB
Handbuch
FRANZIS

Vorwort

USB hat sich in den letzten Jahren zu einer universellen Schnittstelle entwickelt. Neben der PC-Industrie bleibt kaum eine Elektronikbranche von USB verschont. Ob das Autoradio mit Anschluss für USB-MP3-Player, der digitale Satelliten Empfänger für Updates über USB, zur Übermittlung aufgezeichneter Videos oder Bilder auf den PC, oder auch Handys mit MP3 und integrierter Kamera – sie alle besitzen einen USB-Anschluss.

Im Vergleich zu seriellen oder parallelen Schnittstelle eines PC hat USB eine gute Übertragungsgeschwindigkeit und USB-Geräte können während des Betriebes hinzugesteckt oder wieder entfernt werden.

USB steht für *Universal Serial Bus.* Version 1.0 wurde 1995 von einem Konsortium mehrerer großer Elektronik-Unternehmen entwickelt.

Beschäftigt man sich im Detail mit USB, fällt die Komplexität des Themas auf. Wo man früher noch mit der parallelen oder seriellen Schnittstelle des PC die eigene Elektronik einfach steuern und regeln konnte, muss man sich heute mit USB auseinandersetzen, da PCs neuster Generation nur noch mit USB ausgestattet sind. Umfangreiche Literatur findet man im Internet, bei Mikrocontroller-Herstellern und bei den verschiedenen Verlagen. Meist ist in den dort beschriebenen USB-Controllern ein Mikroprozessorkern enthalten. Schnell wird dann das scheinbar einfache Thema „USB“ kompliziert.

Ein Konverter von USB zur seriellen und parallelen Schnittstelle war mein erster Ansatz, mich mit USB- und FTDI-Chips auseinanderzusetzen, da mein neuer PC keine serielle Schnittstelle mehr hatte und ein Mikrocontroller über eine serielle Schnittstelle programmiert werden musste. Schnell stellte ich fest, dass zwar die einfache serielle Übertragung funktionierte, doch das Schalten einzelner Signale der seriellen Schnittstelle im Vergleich zur „normalen“ seriellen (RS232-)Schnittstelle eines alten PC extrem langsam war. Ob USB also doch langsam ist, interessierte mich – und daraus entstand letztendlich dieses Lernpaket mit einfachen Beispielen.

Dieses Lernpaket soll nicht auf jedes Detail aller möglichen USB-Funktionalitäten eingehen. Der in diesem Lernpaket verwendete FTDI-USB-Baustein ist ohne integrierten Mikrocontroller und wird Ihnen zeigen, wie interessante USB-Steuerungen

oder USB-Datenerfassungssysteme zum Teil auch gänzlich ohne Mikrocontroller aufgebaut werden können.

Im täglichen Leben gibt es viele Situationen, die man elektronisch mithilfe eines Computers meistern kann. Das USB-Lernpaket startet mit einfachen LED-An-, LED-Aus- und Ampel-Versuchen über USB. Ein Beispiel einer Alarmanlage und wie man den Wasserstand in einem Aquarium überwachen kann, vertiefen das gelernte Wissen. Auch der Spaß soll nicht zu kurz kommen: Haben Sie noch irgendwo ein Quarzuhrwerk, das, nach einem kleinen Umbau, über USB zum Flaschendrehspiel mutieren kann? Messungen von Helligkeit oder Temperatur mit einem selbst gebauten Analog/Digitalwandler vermitteln weitere Grundlagen, die sowohl für die Software, USB als auch für die Hardware notwendig sind. Manch einer wird sich danach vielleicht wundern, wie einfach ein selbst gebauter A/D-Wandler mit ein paar Zeilen in der Anwendungssoftware behandelt werden kann. Ein kleiner Fernbedienungstester mit Fotodiode, ein Multimeter, ein Kennwort-Datenspeicher und die Verwendung als USB-Dongle sind weitere praktische Beispiele.

Da der Lernadapter auch als USB-Seriell-Konverter für Mikrocontroller Anwendungen dienen kann, werden zusätzliche Softwarebeispiele für die Verwendung mit serieller Kommunikation aufgeführt. Sie werden sehen, wie man die Ein- und Ausgabeleitungen auch für komplexe Schnittstellen wie I^2C oder SPI verwenden kann. Die Umsetzung einer USB-I^2C-Schnittstelle wird zum Kinderspiel, da ein Software-Logikanalyzer das Daten- und Taktsignal im tatsächlich erzeugten Timing-Diagramm visualisiert. Sie können in 5 einfachen Schritten eine universelle I^2C-Schnittstelle entwickeln.

Das letzte Software-Beispiel in diesem Lernpaket zeigt Ihnen in Ansätzen, wie man komplexe Aufgaben, wie z. B. eine Flash-Programmierung des AT89LP-Mikrocontrollers über die SPI/ISP-Schnittstelle, bewerkstelligen kann. Wer bereits Mikrocontrollerelektronik entwickelt, aufgebaut und programmiert hat, kann den Adapter in diesem Lernpaket auch direkt als USB-Seriell-Konverter oder SPI/RS232-Programmieradapter nutzen.

Die einzelnen Beispiele werden detailliert erläutert und sind umfangreich illustriert, damit die Zusammenhänge von der Visual-Basic-Software zur Hardware und in Zusammenhang mit dem verwendeten USB-Baustein FT232R nachvollzogen und für eigene Ideen verwendet werden können.

Wer animiert wurde und Lust hat, einige weitere Schaltungsbeispiele dieses Pakets in die Praxis umzusetzen, sollte einen Lötkolben besitzen. Nun aber erst einmal viel Spaß bei der Lektüre des Buchs.

Dipl. Ing. Jürgen Hulzebosch

Allgemeine Hinweise

Dieses Dokument wurde mit der nötigen Sorgfalt erarbeitet. Es wird keinerlei Haftung für technische und drucktechnische Fehler oder Auslassungen in diesem Dokument übernommen. Außerdem wird keinerlei Haftung übernommen für Schäden, die sich durch die im Dokument enthaltene Information ergeben.

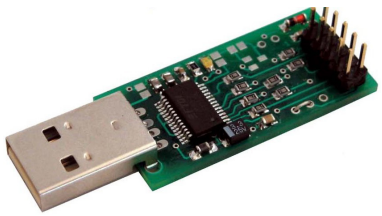
USB 2.0 – erforderlich!
für Windows XP/2000/Vista!

Aktuelle Informationen finden Sie im Internet: www.minimikro.de.

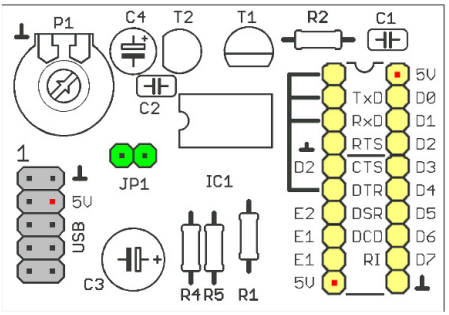
Inhalt

1 Materialliste und die Beschreibungen einzelner Bauelemente

1 × USB-Adapter in SMD-Ausführung



1 × Zusatzkarte



Bauteile:

1 × Duo-LED 5 mm (2 polig)
3 × LED 5 mm/1 × rot, 1 × grün, 1 × gelb
1 × LDR
1 × Fotodiode BPW 40
1 × NTC 0,2–4K7 (4700)
1 × Widerstand 1K (Ringfarben: braun-schwarz-rot)
1 × Widerstand 10K (Ringfarben: braun-schwarz-orange)
1 × Widerstand 100K (Ringfarben: braun-schwarz-gelb)
1 × Widerstand 27 Ohm (Ringfarben: rot-violett-schwarz)

Hinweis: Diese Widerstände können auch in Metallfilmausführung mit anderen Farbringen sein:

1 × Widerstand 1K (Metallfilm, Ringfarben: braun-schwarz-schwarz-braun)
1 × Widerstand 10K (Metallfilm, Ringfarben: braun-schwarz-schwarz-rot)
1 × Widerstand 100K (Metallfilm, Ringfarben: braun-schwarz-schwarz-orange)
1 × Widerstand 27 Ohm (Metallfilm, Ringfarben: rot-violett-schwarz-gold)
1 × Elko 100 µF
1 × 24C16 EEPROM
1 × Diode BAT 42
1 × Anschlussdraht
1 × Verbindung 10-polig von USB-Adapter zur Zusatzkarte
1 × Jumper
1 × CD-ROM mit Dokumentation in PDF-Format, Beispielprogrammen und wichtigen Datenblättern

2 Messen, Steuern und Regeln mit PC-Schnittstellen

Um den PC für eigene Elektronikexperimente zu verwenden, wird eine Schnittstelle zur Außenwelt bzw. zur eigenen Elektronik benötigt:

Über diese Schnittstelle kommuniziert der PC bzw. die PC-Software mit der angeschlossenen Elektronik. In vielen Fällen wurden und werden die seriellen oder parallelen Schnittstellen des PC verwendet. Wie die Namensgebung es schon verrät: Bei der seriellen Schnittstelle werden die Daten seriell übermittelt, d. h., die kleinsten Informationseinheiten werden über eine Signalleitung nacheinander übertragen. Dabei benutzen die übermittelten Daten bei der seriellen Schnittstelle des PC zum Senden und Empfangen jeweils eine getrennte Datenleitung. Zusätzlich gibt es bei der seriellen Schnittstelle des PC weitere Steuersignale, die die Kommunikation vom Computer zur Elektronik und umgekehrt regeln:

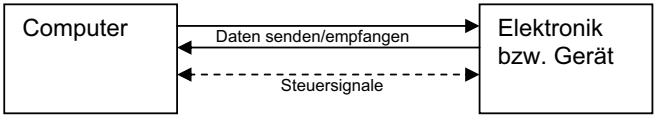


Abb. 2.1: Prinzip der seriellen Schnittstelle

Typische Geräte, die an die serielle Schnittstelle angeschlossen werden, sind Modems, Messgeräte oder auch eine serielle Maus. Es werden für die Verbindung 9- oder 25-polige D-Sub-Steckverbinder verwendet. Auch USB verwendet grundsätzlich eine serielle Datenübertragung.

Bei der parallelen Schnittstelle können bis zu 8 Zustände gleichzeitig auf 8 Datenleitungen übermittelt werden. Wie bei der seriellen, gibt es auch bei der parallelen Schnittstelle zusätzliche Steuersignale.

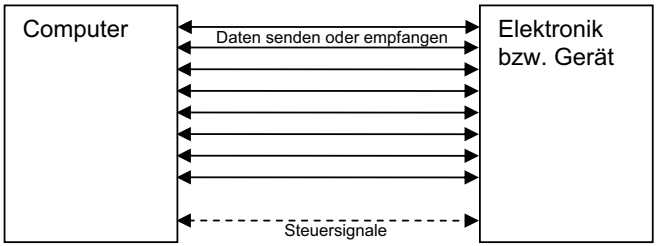


Abb. 2.2: Prinzip der parallelen Schnittstelle

Die parallele Schnittstelle wurde vor allen Dingen bei Druckern und für sonstige Erweiterungen eingesetzt. Für die Verbindung werden 25-polige D-Sub-Steckverbindungen, bzw. bei älteren Geräten Centronics-Stecker, eingesetzt.

An die serielle oder parallele Schnittstelle eines PC wird üblicherweise ein Gerät angeschlossen. Hat man einen Drucker an eine parallele Schnittstelle angeschlossen, benötigt man eine weitere parallele Schnittstelle für einen zweiten Drucker. Sind nicht genügend Schnittstellen frei (oder nicht vorhanden), kann man den PC mit Erweiterungskarten für die verschiedensten, internen Steckkartenplätze (Slots) aufrüsten. Bedingt durch frühere Definitionen bei Hardware-, Software- und Betriebssystementwicklungen bei DOS und Windows basierenden PCs werden bis Windows XP im Allgemeinen 2 parallele und bis zu 32 serielle Schnittstellen unterstützt. Mit spezieller Geräteherstellerunterstützung sind bis zu 8 parallele und 256 serielle Schnittstellen möglich.

Es ist notwendig, für die an der seriellen oder parallelen Schnittstelle verwendeten Geräte entsprechende Software – einen Treiber – für ein angeschossenes Gerät zu installieren. Diese erhält man im Allgemeinen vom Gerätehersteller für die jeweiligen Betriebssysteme. Ohne einen solchen Treiber kann das Gerät nicht oder nur eingeschränkt verwendet werden, da es dem Betriebssystem des PC unbekannt bleibt. In der Messtechnik benötigt der Elektroniker für diese Schnittstellen nur selten einen Treiber, da direkt auf Funktionen der Hardware (auf Ein- und Ausgabeadressen) des Computers zugegriffen wird, die seit der Einführung der IBM-XT-PC auch in aktuellen Systemen noch unterstützt werden.

Bei der USB-Entwicklung wurden bestehende Erfahrungen mit an PCs angeschlossen Zusatzgeräten berücksichtigt und in der Anwendung für den Benutzer erheblich vereinfacht. USB steht für *Universal Serial Bus* und ist eine Schnittstelle für viele Geräte. USB kann, ausgehend von einem einzelnen USB-Host-Anschluss, bis zu 127 angeschlossene Geräte ansprechen *(die Adresse 00 ist reserviert)*. Der USB-Host steuert alle Aktivitäten auf den USB-Leitungen und kann jedes Gerät gezielt über eine Adresse ansprechen.

Der Anwendungsbereich wurde, im Vergleich zu den bestehenden seriellen und parallelen Schnittstellen, erheblich erweitert. Nicht nur bei Druckern, Modems, Mäusen, Scannern oder externen Festplatten findet die USB-Schnittstelle Verwendung, sie wird auch bei sonstigen elektronischen Geräten wie MP3-Playern, Handys, Video-Playern, USB-Speichersticks etc. eingesetzt, die oftmals ohne eine Treiberinstallation unmittelbar vom Betriebssystem unterstützt werden. Besitzt der PC nicht schon genügend USB-Schnittstellen, kann man USB einfach mit sogenannten Hubs auf die benötigte Anzahl erweitern.

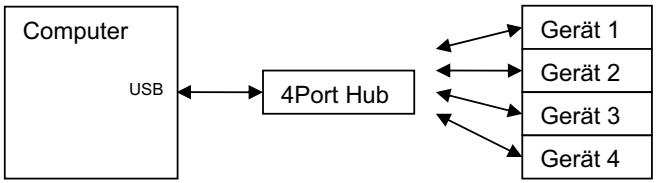


Abb. 2.3: Prinzip USB-Hub

Ausgehend von einem sogennanten USB-Host-Controller (in diesem Beispiel die USB-Schnittstelle des Computers) können an dem nachgeschalteten 4-Port-Hub weitere Geräte sternförmig angeschlossen werden. Ein Hub weist einen oder mehrere USB-Anschlüsse zum Anschluss weiterer USB-Geräte auf. Der Host-Controller im PC regelt die Kommunikation mit den am USB-Host-Controller angeschlossenen USB-Geräten und dem Betriebssystem. Zugleich könnten weitere Hubs an den 4-Port-Hub angeschlossen werden. Gerät 4 könnte zum Beispiel für einen weiteren 7-Port-Hub verwendet werden und somit die Anzahl der USB-Anschlüsse auf 10 erweitern. In diesem Fall spricht man von einer Kaskadierung.

Es kann jeweils immer nur ein angeschlossenes USB-Gerät mit einem USB-Host-Controller kommunizieren. Es besteht keine Möglichkeit für das gleichzeitige Versenden einer Mitteilung (*Broadcasting*) an mehrere angeschlossene Geräte. Um die Bandbreite zu erhöhen, besitzen neuere PCs deshalb oftmals 2,4 oder 8 Host-Controller. Wenn also mehrere USB-Geräte untereinander Daten austauschen sollen, z. B. zum Kopieren von Daten von einer USB-Festplatte auf eine andere und umgekehrt, ist es zweckmäßig, die Festplatten an unterschiedliche, statt am selben USB-Host-Controller anzuschließen.

Der USB-Stecker besitzt nur 4 Anschlussleitungen: neben einem Stromanschluss für die angeschlossenen Geräte, die seriellen und parallelen Schnittstellen fremd sind, nur noch zwei weitere Datenleitungen, über die die gesamte Kommunikation mit dem Host-Controller abgewickelt wird. Damit sind die USB-Steckverbindungen sehr preiswert und konnten miniaturisiert werden.

An USB angeschlossene Geräte werden heute meist automatisch von den Betriebssystemen Windows 2000, XP und Vista erkannt. Schließt man ein Gerät das erste Mal an, wird (meist) zur Installation eines Treibers aufgefordert. Nach erfolgreicher Installation und Einbindung erkennt das Betriebssystem das Gerät beim nächsten Mal automatisch. Oftmals bemerkt man aber die Einbindung des Treibers gar nicht, da das Betriebssystem z. B. für USB-Speichersticks oder USB-Festplatten schon Treiber in einer Hardware-Kompatibilitätsliste bereithält und automatisch einbindet.

2.1 Informationseinheiten – Daten

Was versteht man beim Computer bzw. in der Informatik unter dem Begriff Daten? Wie sehen Daten aus, die in Computern, bzw. Mikrocontrollern zur Verarbeitung genutzt oder auch über die verschiedenen Schnittstellen von einer Elektronik zum Computer, oder im Computer selbst, verwendet werden? Die Antwort im Internet:

[wikipedia]: „Daten sind logisch gruppierte Informationseinheiten, die zwischen (Computer-)Systemen übertragen werden. Sie bestehen aus einem Binärcode. Man benötigt Codiervorschriften, mittels derer Informationen in Daten transformiert werden können. In der Informatik und Datenverarbeitung (EDV) versteht man Daten als (maschinen-)lesbare und bearbeitbare in der Regel digitale Repräsentation von Information. Die Information wird dazu meist zunächst in Zeichen (bzw. Zeichenketten) codiert, deren Aufbau strengen Regeln folgt."

„Daten" ist die Mehrzahl von Datum, der kleinsten, unteilbaren Information in einer Informationseinheit (Nachricht), die übermittelt wird. Die Nachricht hat eine Bedeutung für den (oder die) Empfänger und folgt festgesetzten Regeln. Bei einem digital arbeitenden Computer sind die kleinsten Informationseinheiten die Werte 0 und 1. Durch das Aneinanderreihen von Nullen und Einsen entstehen die Zeichenketten und Zahlen für die weitere Verarbeitung in Soft- und Hardware innerhalb des Computers. (Computer wird aus dem engl. Wort to „compute" = rechnen, bzw. zählen, abgeleitet.)

Die einfachste Form einer *analogen* Datenübermittlung ist vielleicht das Telefon. Mit dem Sprechen in die Sprachmuschel werden Schallwellen als elektrische Information erzeugt, diese übertragen und beim Empfänger über einen Lautsprecher wieder in Schallwellen umgewandelt.

Im Gegensatz zu digitalen Informationen mit den (diskreten) Werten 0 oder 1 kann ein analoges Signal jeden beliebigen Wert innerhalb festgelegter Grenzwerte einnehmen, also z. B. auch 0,1–0,2–0,2345 ...

2.2 Binär zählen (Logische 0 und 1)

Die Signale bzw. die Signalleitungen an einer digitalen Schnittstelle sind *binär* codiert und können nur die Zustände 0 oder 1 annehmen. Das ist eigentlich ganz einfach, wenn man sich vor Augen hält, wie wir in unserem dezimalen Zahlensystem mit den 10 Zahlen von 0 bis 9 rechnen. Die folgende Gegenüberstellung zeigt es:

Dezimal Basis 10 (0–9)	Binär Basis 2 (0,1)	Hex Basis 16 (0-F)
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10

Bei allen Zahlensystemen erfolgt ein Aneinanderreihen der jeweils diskreten Informationseinheiten, wobei der jeweilige Wert von der Position innerhalb der Information abhängt:

14 Dezimal = 14 $= 1 * 10^1 + 4 * 10^0 = 10 + 4$
14 Binär = 1110 $= 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0 = 8 + 4 + 2 + 0$
14 Hex = E $= 14 * 16^0 = 14$

Der Unterschied in den aufgeführten Zahlensystemen ist nur die Basis bzw. die Anzahl der möglichen, diskreten Werte der Informationseinheiten. Im dezimalen System sind es die Zahlen 0–9 (zur Basis 10, 10 mögliche diskrete Werte) und im binären System beiden Werte 0 und 1 (zur Basis 2).

2.3 Russisches Bauernrechnen

Eine Binärzahl durch 2 zu teilen ist einfach. Man braucht alle Bits nur um eine Stelle nach rechts zu verschieben und das letzte Bit fallen zu lassen. Aus der Zahl 10, binär = 1010, wird dann binär 101 = 5. (Auch aus der 11 wird eine 5). Multiplizieren mit 2 geht durch Schieben der Bits nach links.

Wer Spaß am „Computen“ hat, kann folgende Multiplikation zweier Zahlen testen und versuchen zu ergründen. Diese Art der Multiplikation ist unter dem Begriff „russisches Bauernrechnen“ bekannt:

Beispiel: Multiplikation von 17*12

Die linke Spalte (17) wird immer durch 2 geteilt und die rechte Spalte (12) immer mit 2 multipliziert. Reste sind dabei unbekannt:

17	12
8	24
4	48
2	96
1	192

Jetzt werden nur noch die Zahlen in der Spalte auf der rechten Seite addiert, die in der linken Spalte einen ungeraden Wert besitzen, also 12+192=204. Passt! ... Immer?

2.4 Bits und Bytes

Ein Bit (eng. *Binary* digit) kann die logischen Zustände 0 und 1 annehmen. Reiht man 8 Bit, wie im vorherigen Beispiel, aneinander, kann man mit 8 Bits bis 255 zählen:

1 1 1 1 1 1 1 1 = 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255

8 Bits bilden Byte. Ein Byte kann 256 Wertigkeiten einnehmen (0–255). Ein Byte ist ein Datentyp mit 8 Bit und bildet eine Informationseinheit. Das erste Bit (ganz links = 128) wird auch mit *MSB* (für Most Significant Bit) und das letzte Bit (ganz rechts = 1) mit *LSB* (für Least Significant Bit) bezeichnet.

Ein Word ist ein Datentyp (eine Informationseinheit) mit 16 Bit bzw. 2 Byte und kann 65.536 Wertigkeiten (0–65535) einnehmen.

Ein Megabyte (1 Mega = 1.000.000 Einheiten) sind 10^6 Byte. Manchmal sind damit auch 2^{20} Byte gemeint, was einen Unterschied von nahezu 5 % bedeutet.

3 USB-Grundlagen

3.1 USB 1.0, 1.1, 2.0, On-The-Go

USB 1.0

Der heute unter dem Namen USB 1.0 bekannte *serielle Bus* wurde von Intel entwickelt. Die USB-1.0-Spezifikationen wurden Januar 1996 veröffentlicht. USB 1.0 hatte eine Bitrate von maximal 1,5 MBit/s (Low Speed) und war damit für Massenspeichergeräte wie Festplatten nicht brauchbar. Grundsätzlich wurden Massenspeicher aber schon unterstützt. Die ersten Anwendungen waren z. B. Drucker, Scannen, Mäuse und Tastaturen.

USB 1.1

Mit USB 1.1 wurde es möglich, Datenraten von 12Mbit/s (Full Speed) zu erreichen.

USB 2.0

Im Jahr 2000 wurde USB 2.0, mit einer High-Speed-Transferrate (480 Mbit/s), veröffentlicht und sorgte dafür, dass USB für Peripheriegeräte wie Drucker, Festplatten, Speichersticks etc. wesentlich attraktiver wurde. USB 2.0 ist mit 40-mal höheren Transferraten als im Full-Speed-Modus zu USB 1.X abwärts kompatibel geblieben.

USB On-The-Go

Durch USB On-The-Go (OTG) können entsprechend ausgerüstete USB-Geräte direkt miteinander kommunizieren. Damit kann auf einen Computer, der die Host-Funktion übernimmt, verzichtet werden. Zwei USB-Geräte können ohne einen Host Nachrichten austauschen. Dies ist zum Beispiel bei Kameras mit USB-Anschluss sinnvoll, die direkt einen USB-Drucker steuern möchten, um ein Foto direkt auf einen Drucker (ohne angeschlossenen Computer) auszudrucken.

Wireless USB

Bei dieser USB-Erweiterung handelt es sich um drahtloses USB. Diese Spezifikation wurde im Mai 2005 veröffentlicht und ermöglicht die drahtlose Kommunikation zwischen USB-Geräten bei einer Bruttotransferrate von maximal 480 MBit/s.

3.2 Strom, Spannung- und Geräte-Identifikation der USB-Schnittstelle

Der physikalische USB-Anschluss besitzt vier Leitungen: zwei für die Stromleitungen (Masse und +5 Volt) und zwei Datenleitungen (D+ und D–).

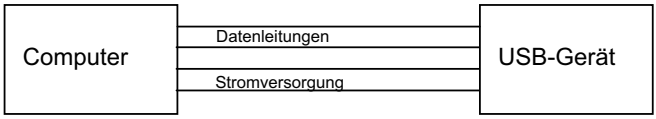


Abb. 3.1: USB-Leitungen

Der Stromverbrauch eines angeschlossenen USB-Geräts darf 100 mA nicht überschreiten, wenn es nicht konfiguriert wurde. Der maximale Stromverbrauch beträgt bei konfigurierten USB-Geräten max. 500 mA. Der Wunsch nach mehr als 100 mA muss vom USB-Gerät angefordert werden. Geräte, die ihren Strombedarf direkt über USB abdecken, werden mit der Kennung *Bus powered* versehen. Natürlich können USB-Geräte auch mit einer eigenen Stromversorgung ausgestattet sein. Diese Geräte besitzen das Merkmal *Self Powered*.

Die Datenleitungen sind zur Optimierung der Datenübertragungsgeschwindigkeit differenziell ausgelegt. Die einzelnen Signalzustände bei diesen Datenleitungen werden hier nicht weiter aufgeführt. Sie wären nur dann wichtig, wenn man z. B. einen USB-Chip entwickeln würde.

Wird ein USB-Gerät an einen USB-Controller angeschlossen oder wieder entfernt, werden die Datenleitungen für die USB-Plug-and-Play-Fähigkeiten verwendet. Besitzt die Datenleitung D+ eines angeschlossenen USB-Geräts z. B. einen Widerstand von ca. 1,5 kOhm zur Versorgungsspannung, wird das Gerät als Full- oder High-Speed-Gerät eingebunden. Besitzt das Gerät hingegen einen 1,5-k-Ohm-Widerstand nach D-, ist es ein Low-Speed-Gerät. Wenn beide Datenleitungen D+ und D- länger als 2,5 µs auf Low-Pegel verbleiben, wird das Gerät als disconnect erkannt und entfernt.

Die USB-Kabellänge darf 5 m nicht überschreiten. Benötigt man eine größere Kabellänge, muss alle 5 m ein USB-Hub verwendet werden.

3.3 Serielle Datenübertragungen USB-RS232-SPI-I²C

Eine Gegenüberstellung häufig zu findender, serieller Schnittstellen:

	USB 2.0	RS232	SPI	I²C
Anz. Geräte	127	2	8	40
Übertragungsrate	bis 480 Mbit/s	bis 115 Kbits/s	bis 2 Mbit/s	bis 2 Mbit/s
Max. Entfernung	5 m, max. 30 m	bis 30 m	bis 3 m	bis 5 m

USB dient zum Anschluss an unterschiedlichste externe Geräte. Die RS232-Schnittstelle findet man z. B. bei Modems, Mäusen und Messgeräten. Die SPI und I²C-Schnittstellen dienen hauptsächlich zur internen Kommunikation von Mikrocontrollern mit weiteren Peripheriebausteinen und unterstützen daher nur kurze Entfernungen.

Bei der seriellen Nachrichtenübermittlung wird eine Nachricht wieder in ihre kleinsten Bestandteile, ein Byte also wieder in 8 Bits zerlegt. Dazu kommen weitere Informationen, wie z. B. Beginn und Ende einer Informationseinheit, welches angeschlossene Gerät angesprochen werden soll oder zusätzliche Informationen, wenn ein angeschlossenes Gerät nicht antwortet oder die Nachricht unvollständig ist.

Die Datentransferrate ist eine Frequenzangabe. Diese errechnet sich bei einer Breite t eines einzelnen Bits aus f=1/t. Bei USB 1.1 und Fullspeed-Mode beträgt die Transferrate 12 MBits/s. Ein einzelnes Bit hat damit eine Zeitdauer von $1/12*10^6$=83 ns. Die Transferrate von 12 MBits/s wird, in Bezug auf die zu übermittelten Rohdaten, nicht erreicht. Die effektiven Werte liegen bei USB 1.1 zwischen 6,5 und 9,5 MBits pro Sekunde, je nachdem, welcher USB-Transfertype für die Nachricht verwendet wird.

3.4 USB – unbekannter serieller Bus

USB wird im Internet mitunter auch als „unbekannter serieller Bus“ bezeichnet. Wenn man sich im Detail mit USB, den Abläufen bei USB-Transfer-Protokollen, Descriptoren, der Datencodierung und Decodierung, der Synchronisation, den Transferarten, den notwendigen (oder auch nicht notwendigen) Treibern auf Betriebssystemebenen, mit den Spezifikationen wie OHCI, UHCI, EHCI oder der Anbieterkennung auseinandersetzt, wird es für den Hobby-Elektroniker schnell undurchsichtig.

Bei USB können Anwendungen, im Gegensatz zu Anwendungen für die serielle oder parallele Schnittstelle, nicht einfach Daten an bestimmte Ein- und Ausgabe-

adressen schreiben oder von diesen lesen. Um auf ein USB-Gerät zuzugreifen, müssen Anwendungen mit einem (Klassen- oder Geräte-)Treiber kommunizieren, der wiederum auf niedrigerer Ebene mit den USB-Treibern kommuniziert, die die Nachrichten über die USB-Datenleitungen steuern und verwalten. Im Gerät müssen die Protokolle implementiert sein, mit deren Hilfe der Computer das Gerät erkennen, identifizieren und damit kommunizieren kann.

Glücklicherweise haben sich damit schon viele USB-Controller-Hersteller auseinandergesetzt. Der in diesem Lernpaket verwendete FTDI-Controller kann man daher auch einfach wie folgt sehen:

1. USB rein → 8 bit parallel oder seriell rein/raus,
2. Das Modul aus diesem Lernpaket einstecken, FTDI-Treiber installieren und für Steuerungen mit einfachen Programmen, wie z. B. Visual Basic oder Pascal, oder für Profis auch mit C, einsetzen.

Interessant ist dabei, dass man die eigentlichen Signale der seriellen Schnittstelle des FTDI-Controllers auch für einen 8-bit-parallelen Datentransfer nutzen kann. Damit können zum Beispiel andere, serielle Schnittstellen, wie z. B. die SPI oder I^2C-Schnittstelle, nachgebildet werden.

Der verwendete FTDI-USB-Controller FT232R ist für USB 2.0 ausgelegt. Der USB-Datentransfer erfolgt innerhalb von 1 ms Frames. Dieses kann man sich einfach als periodisches 1-ms-Zeitfenster vorstellen. Der Millisekunden-Zeittakt wird durch den USB-Host-Controller erzeugt. Innerhalb der Zeitfenster werden die Nachrichten in Datenpaketen übermittelt. Der Datentransfer innerhalb der 1-ms-Frames ist unterschiedlich. Bei USB-Low-Speed beträgt die Datentransferrate bis 1.5 MBits/s, bei USB-Full-Speed die Datentransferrate bis 12 MBits/s, und bei USB-High-Speed (nur USB 2.0) bis 480 MBits/s. Bei USB-High-Speed wird jedes 1-ms-Frame in 8 weitere (Micro) Frames mit je 125 µs unterteilt, um die hohe Datentransferrate zu erreichen.

3.5 USB-Transfertypen

USB-Geräte können in unterschiedlichen Arten mit dem USB-Host-Controllern kommunizieren. Es gibt vier unterschiedliche Transfertypen der Nachrichtenübermittlung, die im Folgenden kurz beschrieben werden:

1. Control-Transfer
   Control-Transfers werden generell in beide Richtungen bestätigt, sodass sich Sender und Empfänger immer sicher sind, dass die Daten angekommen sind. Jedes USB-Gerät muss den Control-Transfer unterstützen. Control-Transfers

dienen z. B. dazu, sich über die Fähigkeiten des USB-Geräts zu informieren und diese zu konfigurieren. Sie sind zum Austausch der ersten Kommunikation elementar wichtig.

2. Interrupt-Transfer
   Interrupt-Transfers arbeiten mit Fehlerprüfung und sind für kleine Datenmengen gedacht. Es ist keine genau festgelegte Transferrate garantiert. Das angeschlossene USB-Gerät wird periodisch abgefragt. Der Abfragetakt liegt bei Low-Speed zwischen 10 und 255 ms, bei Full-Speed zwischen 1 und 255 ms und bei High-Speed bei 125 µs. Bei Low-Speed können pro Abfrage bis zu 8 Byte, bei Full-Speed bis zu 64 Byte und bei High-Speed bis zu 1.024 Byte übertragen werden. Daraus ergeben sich maximale Datenraten von 800 Byte/s bei Low-Speed, 64 kByte/s bei Full-Speed und bis zu 24 MByte/s bei High-Speed. Der Interrupt-Transfer hat, trotz der Namensgleichheit, nichts mit Prozessor-Interrupts zu tun.
3. Bulk-Transfer
   Bulk-Transfers sind für größere Datenmengen die eine Fehlerprüfung benötigen, aber nicht zeitkritisch sind. Diese Transfers besitzen eine niedrige Priorität. Sie werden vom Controller durchgeführt, wenn andere Isochrone und Interrupt-Transfers abgeschlossen sind. Der verwendete FTDI-Chip FT232R unterstützt diesen Transfertyp.
4. Isochronus-Transfer
   Isochrone Transfers sind für USB-Geräte, die große Datenmengen verarbeiten, und haben eine garantierte Datenrate. Bei isochronen Transfers erfolgt keine Fehlerkorrektur! Der Isochronus-Transfer steht bei Low-Speed nicht zur Verfügung.

3.6 FTDI-USB-Treiber

Ein Treiber ist im Allgemeinen das Bindeglied zwischen Hardware und Software. Um also auf ein USB-Gerät zugreifen zu können, muss ein Programm mit einem Treiber kommunizieren, der wiederum auf niedrigerer Ebene mit den USB-Treibern kommuniziert, die die Nachrichten über die USB-Datenleitungen steuern. Prinzipiell wäre es auch möglich, die niedrigere Ebene unmittelbar mit dem eigenen Programm anzusprechen, doch dann müsste man sich vielfach im Detail mit den herstellerspezifischen Eigenschaften des USB-Geräts auseinandersetzen. Im Allgemeinen besitzen, bei den neueren Windows-Betriebssystemen, heute nur die Treiber die notwendigen Zugriffsrechte, um eine Computer-Hardware anzusprechen.

FTDI liefert für den verwendeten FT232R-USB-Controller für Windows zwei unterschiedliche Treiber. Der erste Treiber ist ein sogenannter *VCP*(Virtueller

COM-Port)-Treiber. Dieser Treiber sorgt dafür, dass das Betriebssystem einen seriellen COM-Port einbindet, wenn der FTI-USB-Controller über USB mit dem Computer verbunden wird. Damit wird eine serielle Standardschnittstelle emuliert, dem Betriebssystem also ein (weiterer) COM-Port vorgetäuscht. So kann man schnell neuere Computer mit einer virtuellen, seriellen Schnittstelle nachrüsten und hohe Kompatibilität mit bestehenden Anwendungsprogrammen erreichen.

Der zweite Treiber wird von FTDI *D2XX-Treiber* genannt. Dabei handelt es sich um einen Treiber (FTD2XX.SYS) für das WIN32 Driver Model (WDM). Die eigene Anwendungssoftware kommuniziert über diesen Treiber mit dem Windows USB-Treiber-Stack, der wiederum auf das USB-Gerät zugreift.

Zu beiden Treibern gibt es bei FTDI im Internet unter www.ftdichip.com entsprechende Programmierunterlagen.

Bis Anfang 2007 musste man sich bei der Treiberinstallation noch für einen der beiden Treiber entscheiden. Wollte man den zweiten Treiber verwenden, musste man den ersten deinstallieren. Dies hat sich mit der Treiberversion 2.0 in 2007 geändert. Ein Treiber für beide Varianten (D2XX und VCP) erspart lästige De- und Neuinstallationen, wenn man experimentieren möchte.

Bei FDTI gibt es nicht nur Treiberunterstützung für alle gängigen Windows-Varianten, sondern auch für Linux, Mac OS, Windows CE etc.

3.7 USB-Anbieter-Identifizierung

Damit ein USB-Gerät auf den Markt gebracht werden darf, werden eine sogenannte Vendor *ID* (VID) und eine *Produkt ID* (PID) benötigt. Die Vendor ID wird gegen eine Lizenzgebühr vom USB-Konsortium vergeben. Mit der Verwendung des FTDI-Controllers darf kostenlos die FTDI-Vendor-ID verwendet werden. Verwendet man den FTDI-Controller für andere Zwecke, kann man von FTDI auch eine andere PID-Nummer erhalten.

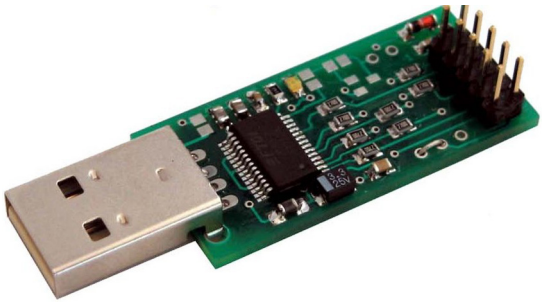
Die VID- und PID-Nummern werden innerhalb des FTDI-Controllers in einem EEPROM gespeichert und könnten sogar verändert werden. In diesem Fall müssen auch die Treiberinstallationsroutinen angepasst werden. Nach Änderungen ist dann im Allgemeinen eine Neuinstallation des angepassten Treibers erforderlich.

Für das Lernpaket werden die FTDI-VID und die FDTI-PID verwendet, damit jederzeit ein aktualisierter Treiber aus dem Internet von www.ftdichip.com eingesetzt werden kann.

4 Hardware USB-Zusatzplatine

4.1 Hardware USB-Adapter und Beschreibung

Die Hardware in diesem Lernpaket wurde ursprünglich für die serielle SPI-Programmierung der Atmel-AT89LP-Mikrocontroller-Familie entwickelt. Literatur dazu finden Sie bei www.minimikro.de und bei www.atmel.com.



Der Adapter besitzt auf der Ausgangsseite einen 10-poligen Anschluss, der, je nach Verwendung, folgendermaßen belegt ist:

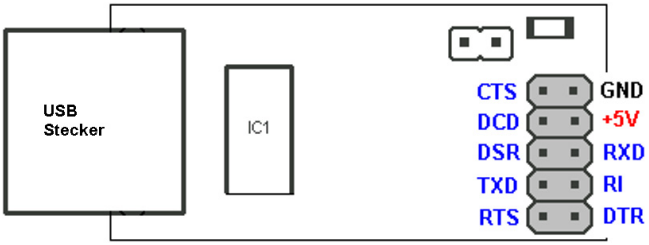


Abb. 4.1: Steckerbelegung serielle Signalleitungen

Pin 1 und 2 (= GND), liegen, von oben gesehen und den USB-Stecker vom Körper weg gerichtet, auf der rechten Seite. Pin 1 in Richtung zum USB-Controller-Chip und Pin 2 in Richtung zum unteren Platinenrand.

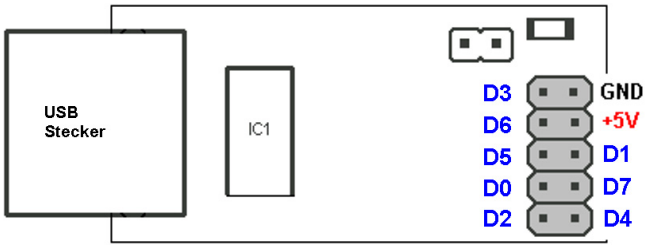


Abb. 4.2: Steckerbelegung parallele Datenausgabe (BitBang-Mode)

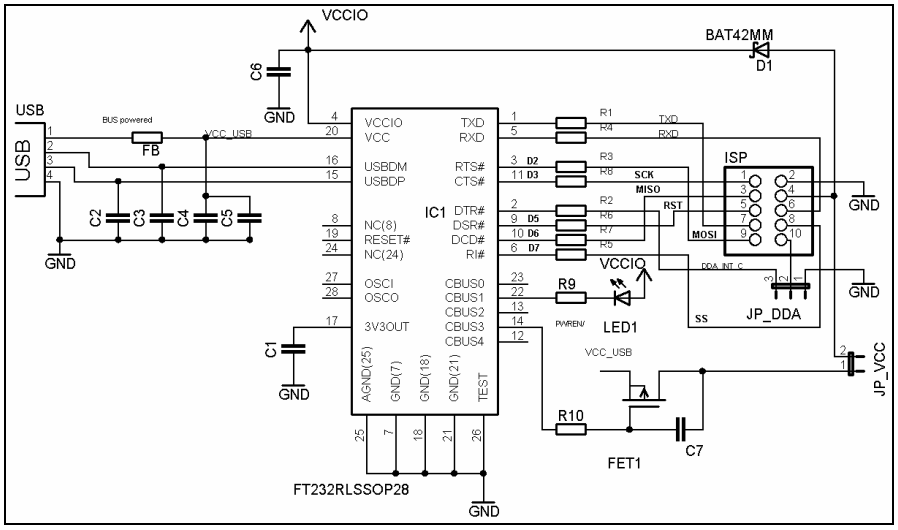


Abb. 4.3: Schaltplan des USB-Adapters

Die Beschaltung des FT232R ist einfach. Die Widerstände R1 bis R8 in den Signalleitungen betragen 256 Ohm und reduzieren den Strom bei einem möglichen Kurzschluss. Über den FET1 und den Jumper JP_Vcc wird die 5-Volt-USB-Spannung an Pin 4 des Steckers geleitet. Sobald der angeschlossene Computer in den Stand-by-Mode geschaltet wird, ist mit dem FET1 sichergestellt, dass die USB-Konventionen eingehalten werden und der Stromverbrauch über USB entsprechend gering ist. R10 und C7 sorgen dafür, dass ein sanftes Einschalten erfolgt. C4 und C6 besitzen jeweils eine Kapazität von 100 nF; C5 hat einen Wert von 3.3 µF. C3 und C4 (jeweils 47 pF) dienen der Reduzierung von Störeinflüssen.

Der Ferritkern (FB) in der Vcc_USB-Versorgungsspannung ist eine FTDI-Empfehlung und wird bei Störanfälligkeiten benötigt. Pin 10 des ISP-Pfostensteckers wird über den Jumper JP_DDA auf D4 gelegt und ist fest verdrahtet.

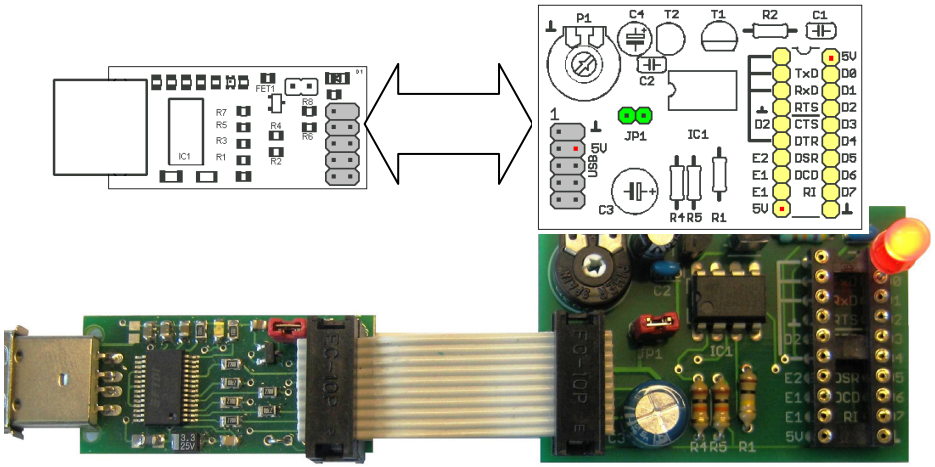
Beim Experimentieren mit dem Lernpaket sollte nur der USB-Adapter des Lernpakets an einem USB-2.0-Port und ohne USB-Hub angeschlossen sein!

Fassen Sie die USB-Platine möglichst immer nur an den Rändern der Platine an. Sie können sonst ggf. den USB-Lernadapter bereits durch statische Entladungen zerstören!

Wenn Sie den Adapter einstecken, werden Sie nur ein kurzes Aufleuchten der LED1 sehen.

4.2 Hardware Zusatzplatine

Auf der Zusatzplatine sind die meisten Bauelemente für weitere Experimente platziert. Die Platine wird direkt mit dem USB-Adapter verbunden. Der IC Sockel für die 8 I/O Signalleitungen dient zugleich als kleines Steckbrett.



Auf der Zusatzplatine befindet sich ein Operationsverstärker und Bauelemente für einen einfachen A/D-Wandler. Erläuterungen zum A/D-Wandler finden Sie in Kapitel 11.2. Mit dem Jumper JP1 auf der Karte können Sie die Versorgungsspannung für den analogen Teil deaktivieren. Die Pins 1,2,3 und 6 sind beim Stecksockel direkt miteinander verbunden.

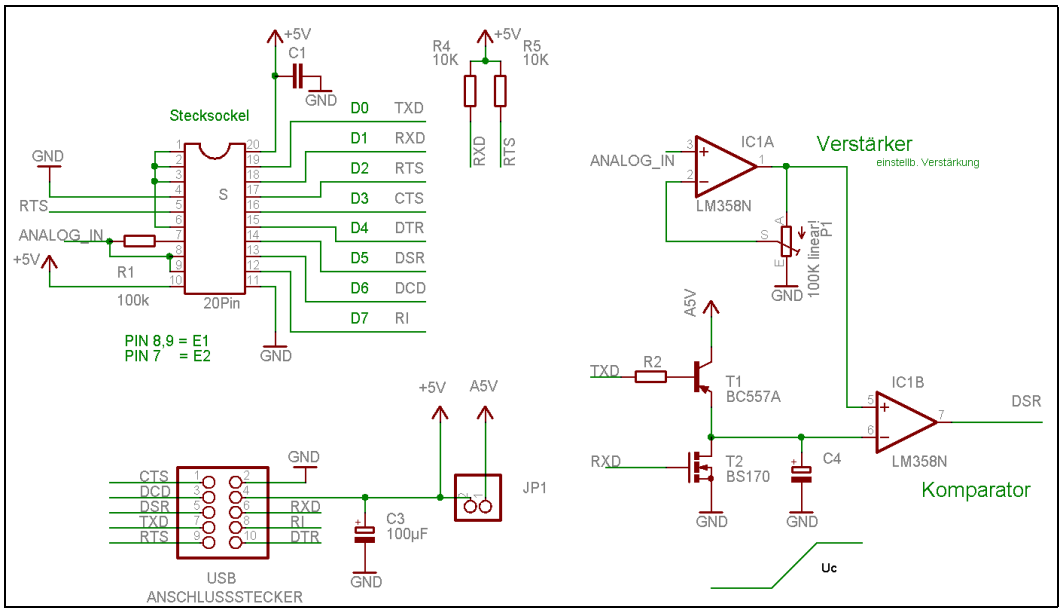


Abb. 4.4: Schaltbild Zusatzkarte

Bauteileliste Zusatzkarte:

Bauteil	Wert	
C1	100n	C-EU025–024X044
C2	100n	C-EU025–024X044
C3	100 µF	CPOL-EUE2.5–7
C4	0,47 µF	CPOL-EUE2.5–5
IC1	LM358N	DIL08
JP1	1X02	Pinhead + Jumper
P1	100K	Linear
R1	100k	R-EU_0204/7
R2	39K	R-EU_0204/7
R4	10K	R-EU_0204/7
R5	10K	R-EU_0204/7
S1	20Pin, DIL20	gedrehter Sockel
T1	BC557 A	TO92
T2	BS170	TO
USB	2X05	

Sachverzeichnis